APPENDIX A

1. A method for implementing a functional memory, in which memory data is stored as data units for each of which a dedicated storage space is assigned in the memory, in accordance with which method

- the memory is implemented as a directory structure comprising a tree-shaped hierarchy having nodes at several different levels, wherein an individual node can be (i) a trie node associated with a logical table wherein an individual element may contain a pointer pointing to a lower node in the tree-shaped hierarchy and wherein an individual element may also be empty, in which case the content of the element corresponds to a nil pointer, the number of elements in the table corresponding to a power of two, or (ii) a bucket containing at least one element in such a way that the type of an individual element in the bucket is selected from a group including a data unit, a pointer to a stored data unit, a pointer to another directory structure and another directory structure,

- address computation performed in the directory structure comprises the steps of

- (a) selecting in the node at the uppermost level of the tree-shaped hierarchy a given number of bits from the bit string formed by the search keys employed, forming from the selected bits a search word with which the address of the next node is sought in the node, and proceeding to said node,

- (b) selecting a predetermined number of bits from the unselected bits in the bit string formed by the search keys employed and forming from the selected bits a search word with which the address of a further new node at a lower level is sought from the table of the node that has been accessed,

APPENDIX A

- repeating step (b) until an element containing a nil pointer is encountered or until the address of the new node at a lower level is the address of a bucket,

wherein the nodes to which a given node contains pointers are child nodes of said given node and the nodes to which the child nodes contain pointers are grandchild nodes of said given node,

characterized by

implementing trie nodes as quad nodes of four elements, and replacing in at least part of the directory structure groups of successive nodes by compressed nodes in such a way that

(a) an individual group comprising a given quad node and its child nodes is replaced by a node whose logical table has 16 elements, and

(b) a compressed node known per se is formed from said node of 16 elements by physically storing in the node only non-nil pointers and in addition a bit pattern on the basis of which the physical storage location in the node, corresponding to the search word, can be determined.

2. A method as claimed in claim 1, characterized in that replacement is carried out in the directory structure on all groups in which the quad node has two child nodes.

3. A method as claimed in claim 1, characterized in that replacement is carried out in the directory structure on all groups in which the quad node has eight grandchild nodes at most.

4. (Amended) A method as claimed in claim 2 [or claim 3], characterized in that an upper limit is set for the number of pointers in the

compressed node, wherein when said limit is exceeded the compressed node is again decompressed to a quad node and child nodes.

5. A method as claimed in claim 4, characterized in that eight pointers is employed as said upper limit.

6. A method as claimed in claim 1, characterized in that ten pointers is employed as said upper limit.

7. (Amended) A method as claimed in claim 2 [or claim 3], characterized in that compression is additionally carried out on at least some of the quad nodes (N80...N82) in the structure in such a way that only non-nil pointers are physically stored in the node and in addition a bit pattern (BP2) on the basis of which the physical storage location in the node, corresponding to the search word, can be determined.

8. A method as claimed in claim 1, characterized in that the non-nil pointers are stored in the compressed node in succession in the same order that they have in said table.

9. A method as claimed in claim 8, characterized in that the bit pattern has one bit for each element in the table, each bit indicating whether the corresponding element contains a nil pointer or a non-nil pointer.

10. A method as claimed in claim 8, characterized in that space is reserved for the bit pattern in all trie nodes of the directory structure.

11. A method as claimed in claim 8, characterized in that space is reserved for the bit pattern in the compressed nodes only.

APPENDIX A

12. A method for implementing a functional memory, in which memory data is stored as data units for each of which a dedicated storage space is assigned in the memory, in accordance with which method

- the memory is implemented as a directory structure comprising a tree-shaped hierarchy having nodes at several different hierarchy levels, wherein an individual node can be (i) an internal node associated with a logical table wherein an individual element may contain a pointer pointing to a lower node in the tree-shaped hierarchy and wherein an individual element may also be empty, in which case the content of the node corresponds to a nil pointer, the number of elements in the table corresponding to a power of two, or (ii) a leaf containing an element the type of which is selected from a group including a pointer to a stored data unit, a data unit, and a pointer to a node in another directory structure,

- address computation performed in the directory structure comprises the steps of

- (a) selecting in the node at the uppermost level of the tree-shaped hierarchy a given number of bits from the bit string formed by the search keys employed, forming from the selected bits a search word with which the address of the next node is sought in the node, and proceeding to said node,

- (b) selecting a given number of bits from the unselected bits in the bit string formed by the search keys employed, and forming from the selected bits a search word with which the address of a further new node at a lower level is sought from the table of the node that has been accessed,

- repeating step (b) until an empty element is encountered or until the address of the new node at a lower level is the address of a leaf,

wherein the nodes to which a given node contains pointers are child nodes of said given node and the nodes to which the child nodes contain pointers are grandchild nodes of said given node,

characterized by

implementing internal nodes as quad nodes having four elements, and replacing in at least part of the directory structure groups of successive nodes by compressed nodes in such a way that

- an individual group comprising a given quad node and its child nodes is replaced by a node whose logical table has 16 elements, and

- a compressed node known per se is formed from said node of 16 elements by physically storing in the node only non-nil pointers and in addition a bit pattern on the basis of which the physical storage location in the node, corresponding to the search word, can be determined.

13. A method as claimed in claim 12, characterized in that replacement is carried out in the directory structure on all groups in which the quad node has two child nodes.

14. A method as claimed in claim 12, characterized in that replacement is carried out in the directory structure on all groups in which the quad node has eight grandchild nodes at most.

15. A method as claimed in claim 13 [or claim 14], characterized in that an upper limit is set for the number of pointers in the compressed node, wherein

when said limit is exceeded the compressed node is again decompressed to a quad node and child nodes.

16. A method as claimed in claim 15, characterized in that eight pointers is employed as said upper limit.

17. A method as claimed in claim 15, characterized in that ten pointers is employed as said upper limit.

18. A method as claimed in claim 13 [or claim 14], characterized in that compression is additionally carried out on at least some of the quad nodes in the structure in such a way that only non-nil pointers are physically stored in the node and in addition a bit pattern (BP2) on the basis of which the physical storage location in the node, corresponding to the search word, can be determined.

19. A method as claimed in claim 12, characterized in that the non-nil pointers are stored in the compressed node in succession in the same order that they have in said table.

20. A method as claimed in claim 19, characterized in that the bit pattern has one bit for each element in the table, each bit indicating whether the corresponding element contains a nil pointer or a non-nil pointer.

21. A method as claimed in claim 19, characterized in that space is reserved for the bit pattern in all trie nodes of the directory structure.

22. A method as claimed in claim 19, characterized in that space is reserved for the bit pattern in the compressed nodes only.

23. A memory arrangement for storing data units, said memory arrangement comprising a directory structure in which progress is made by using search words

formed from a bit string constituted by the search keys employed in each case, said directory structure comprising a tree-shaped hierarchy having nodes at several different hierarchy levels, wherein an individual node can be (i) a trie node associated with a logical table wherein an individual element may contain a pointer pointing to a lower node in the tree-shaped hierarchy and wherein an individual element may also be empty, in which case the content of the element corresponds to a nil pointer, the number of elements in the table corresponding to a power of two, or (ii) a bucket containing at least one element in such a way that the type of an individual element in the bucket is selected from a group including a data unit, a pointer to a stored data unit, a pointer to a node in another directory structure and another directory structure,

characterized in that

some of the trie nodes are quad nodes whose logical table has four elements and some are nodes whose logical table has 16 elements and in which only non-nil pointers are physically stored in addition to a bit pattern (BP1) on the basis of which the physical storage location in the node, corresponding to the search word, can be determined.

24. A method as claimed in claim 23, characterized in that at least some of said quad nodes store physically only those pointers that are non-nil pointers and in addition a bit pattern (BP2) on the basis of which the physical storage location in the node, corresponding to the search word, can be determined.

25. A memory arrangement for storing data units, said memory arrangement comprising a directory structure in which progress is made by using search words formed from a bit string constituted by the search keys employed in each case, said

directory structure comprising a tree-shaped hierarchy having nodes at several different hierarchy levels, wherein an individual node can be (i) an internal node associated with a logical table wherein an individual element may contain a pointer pointing to a lower node in the tree-shaped hierarchy and wherein an individual element may also be empty, in which case the content of the element corresponds to a nil pointer, the number of elements in the table corresponding to a power of two, or (ii) a leaf containing at least one element of a type selected from a group including a pointer to a stored data unit and a pointer to a node in another directory structure,

characterized in that

some of the trie nodes are quad nodes whose logical table has four elements and some are nodes whose logical table has 16 elements and in which only non-nil pointers are physically stored in addition to a bit pattern (BP1) on the basis of which the physical storage location in the node, corresponding to the search word, can be determined.

26. A method as claimed in claim 23, characterized in that at least some of said quad nodes store physically only those pointers that are non-nil pointers and in addition a bit pattern (BP2) on the basis of which the physical storage location in the node, corresponding to the search word, can be determined.